USB Ports Find New Homes

by Jon Titus,
Senior Technical Editor

People who have set up PCs know the Universal Serial Bus (USB) approaches plug-and-play performance. So, dropping USB ports into embedded systems might provide an easy way to control external devices. After all, with only two signal lines and two power connections, how difficult can it be to design in a USB port? That task, which sounds simple, will challenge designers. Vendors can help, though, by supplying everything from boards to software.

The easiest way to add USB ports to an embedded system involves using a Windows-based computer. Many single-board computers (Figure 1) offer built-in USB ports and run Windows software. And, USB peripherals come with ready-to-use Windows drivers. The job can get difficult when you need a real-time operating system (RTOS). Several companies provide chips and software that can simplify designing USB ports, and you can also license intellectual property (IP) that adds ports to a system on a chip (SOC).

A USB device can act as either a host or a peripheral. Peripherals require little intelligence, but host USB ports demand processing power and control software. The USB On-the-Go (OTG) specification provides for peripherals with limited host capabilities so USB devices can communicate without needing a PC-based host device. (I'll cover that topic in a future article.)

The latest standard (USB 2.0) offers bus speeds of 1.5 Mbits/sec (low speed), 12 Mbits/sec (full speed) and 480 Mbits/sec (high speed). Unfortunately, many consumers think a device marked "USB 2.0" always operates at the 480 Mbits/sec rate. But many USB 2.0-compatible devices operate at one of the slower transfer rates.

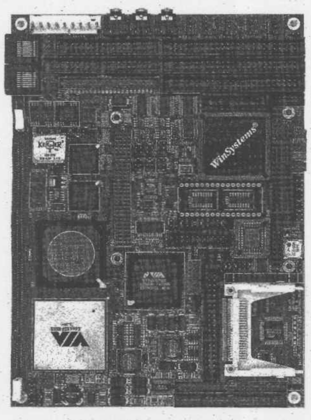

Figure 1. A single-board computer such as the EBC-C3Plus offers built-in USB host ports that operate under control of Windows, Linux or an RTOS that runs on x86 processors. (Courtesy of WinSystems.)

When planning an embedded application, be sure you know what types of USB devices — printers, mass-storage units, and I/O controllers — you plan to employ. Then, determine how many ports you need and whether you will require host, peripheral or combination ports. (Most embedded applications do not need peripheral ports.)

As you evaluate port configurations, also determine your data-rate needs. Although an application may transfer data at an average of 10 Mbits/sec, for example, it may need to transfer bursts of data at higher rates. Keep in mind that higher transfer rates demand larger buffers at each end of a USB connection. You may want to cut performance slightly to reduce the number of expensive buffers that a design needs.

If you're unsure about data rates, buy development boards from USB host-chip vendors and simulate your application on a PC. Host-chip vendors include TransDimensions (Irvine, CA; www.transdimension.com), Cypress Semiconductor (San Jose, CA; www.cypress.com), NEC (New York, NY; www.necus.com) and Philips Semiconductors (Eindhoven, Netherlands; www.semiconductors.philips.com). TransDimension, for example, offers two-port and three-port host chips and software. (Many more companies sell chips that can act as USB peripheral controllers.)

Software support for host-controller chips varies from a reliance on Windows-supplied drivers for the Philips ISP1561, to support for many RTOSs for the TransDimension devices. Cypress provides Linux and VxWorks drivers for its EZ-Host IC. All USB host ports require a "stack," which simply means a layered arrangement of software, as shown in Figure 2. The layers progress from the bottom physical layer (wires and chips) up to the application-software layer on top.

At the physical layer, chip and IP vendors have adopted standard software-interface "connections" that can simplify the use of host devices in an embedded design. Thus, a stack will usually comply with the Open Host Controller Interface (OHCI), the Universal Host Controller Interface (UHCI) or the more-recent Enhanced Host Interface Controller (EHCI) specification developed for USB 2.0. These specifications describe the register-level operations for host controllers. By adopting these standards, chip manufacturers provide a common interface for driver software. The OHCI, UHCI and EHCI specs define the inter-operation of the host-controller registers and the driver software.

Often, a hardware-abstraction layer, or wrapper, shields the stack and drivers from the implementation details of a specific microprocessor. Designers get the wrapper's source code, so they can modify it to operate with their chosen microprocessor and RTOS. If you don't want to get immersed in this level of software detail, suppliers may customize wrappers or recommend consultants who do such work. You may find it easier to start a project with compatible microprocessor, stack, RTOS and USB host controller.

In addition to a stack, developers will need drivers that control peripherals. Although Windows, Windows CE and Linux can draw from a cornucopia of drivers for many peripherals, you won't find the same variety available for other operating systems. IC vendors provide some assistance by offering USB "class drivers" that supply a minimum set of functions to control devices within a class. Individual classes cover printers, cameras, human-interface devices (HIDs), mass storage equipment, communication devices and so on. A printer class driver, for example, can obtain printer status, initialize the printer, send and receive printer data and reset the printer.

So, you can probably find a printer class driver that works with a specific RTOS and processor. A printer manufacturer, though, may offer more capable drivers that build on those supplied in a class driver, often called a class library. But, if you plan to control a custom I/O device, also plan to write your own driver.

Although a PC can store hundreds of USB device drivers, whether or not a user needs them, embedded systems can't offer this flexibility. Thus, you may have to specify only those peripherals guaranteed to work with your system. Due to limited storage space in many embedded systems, furnishing drivers for only a few devices makes sense. Remember, if you can't define the devices you expect to connect to an embedded system, you have a poor design. Before you consider software and drivers, specify the devices your USB ports will communicate with.

In some cases, the need for small size, high reliability or large production

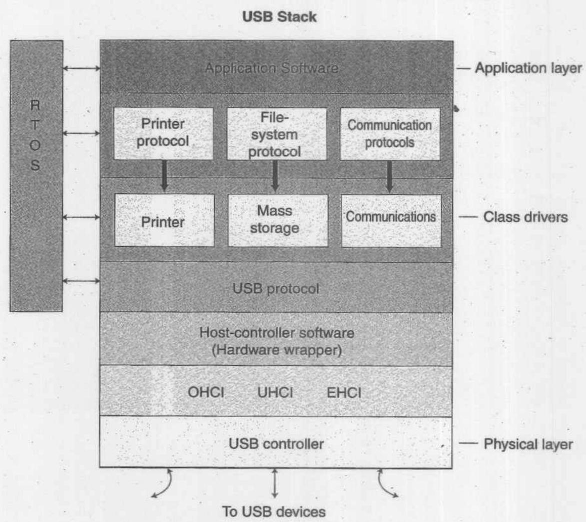

Figure 2. The diagram of the stack for a USB host port shows the arrangement of software layers, from the physical layer at the bottom, to the application software at the top.

volumes may "push" an embedded device from a board into a chip. So, engineers must license intellectual property they can incorporate into ASICs or FPGAs. Several companies supply USB host controllers as IP: Mentor Graphics provides Inventra, a full-speed USB controller that can act as either a host or a peripheral. Synopsys (Mt. View, CA; www.synopsys.com) offers DesignWare cores that provide various types of USB ports. Lastly, ARC International (San Jose, CA; www.arc.com) licenses several USB designs. Companies provide their USB cores either as VHDL or Verilog code.

Fortunately, USB cores come with standard hardware interfaces such as the Advanced High-speed Bus (AHB), developed by ARC International for its microprocessors, and it is available on many other processor cores. IP vendors also may support the Peripheral Virtual Component Interface (PVCI) standardized by the Virtual Socket Interface Alliance (www.vsi.org). If you decide to license USB IP, be sure it will work with your chosen RTOS. IP licensors should supply a list of compatible software products and development tools. And, IP vendors themselves may offer software assistance.

Unfortunately, the scope of USB design considerations goes beyond what one article can cover. The suggested materials below point to additional resources.

Book Review

"Embedded Ethernet and Internet Complete," by Jan Axelson, Lakeview Research, Madison, WI. 2003. $49.95. 482 pages. ISBN: 1-931448-00-0.

More and more equipment includes built-in Ethernet ports that communicate with computers and equipment across a lab bench or across a continent. But without the practical information Axelson provides in her latest book, designing an Ethernet port and getting it to work properly can present engineers with a challenge. So if you plan to design an embedded system with a built-in network connection, buy this book.

The hands-on approach and real software examples make this book invaluable. It serves as a how-to manual for designers and as a reference book for people who need to better understand how networks operate. Readers don't get a lot of useless block diagrams and academic descriptions of networking. Instead, they receive practical information they can immediately apply.

Although experts, and even beginners, may want to skip the first two chapters on basic network information, readers shouldn't take this information for granted. Explanations of a protocol "stack" and basic network hardware, for example, provide a solid foundation for later learning.

Early in the book, Axelson examines commercial OEM network modules that engineers can add to a design or use as reference designs. (Some modules offer more than just Ethernet ports.) Descriptions of these modules, and the development tools that accompany them, provide a great head start for a networking project. But most modules don't provide plug-and-play Ethernet communications. Using these devices requires that engineers understand the Internet Protocol (IP), which the author covers thoroughly in its own chapter. Moving up the protocol stack, the next chapter describes the operation of the User Datagram Protocol (UDP) and the more common Transmission Control Protocol (TCP). Both this chapter and the one on IP supply code snippets. Although code samples cover only C-language software for the Rabbit Semiconductor module and Java code for the Dallas Semiconductor TINI board, the explanations should let readers apply the software concepts to other modules. Don't fret about typing in sample code, though. The author's Web site (www.lvr.com) provides it, along with lists of additional resources, updates and corrections (if any) to the book, and a Windows communication program that readers can use to test network connections.

It's one thing to get hardware working, but yet another to communicate useful information. That may be why the author devoted half this book to network applications such as using Web pages to display dynamic data, interacting with Web pages to obtain data, sending email and using the file-transport protocol (FTP). Each topic gets its own chapter, along with a wealth of code that makes it easy to start work on a prototype application. In the end, readers may find it easy to set up a network connection, but more difficult to decide how to use it effectively. So, these latter chapters offer ways to apply remote control and two-way data communications. Whether you work as a hardware designer or as a software developer, this book will help you better apply network interfaces, even if you never build one.

...Jon Titus

For Further Reading

Axelson, Jan, "USB Complete," 2nd edition, Lakeview Research, Madison, WI. 2001. Lots of practical information centered on PC applications. The author's Web site, www.lvr.com contains much USB information.

Saunders, Mark, "Embedded Considerations for USB On-the-Go," Mentor Graphics, Wilsonville, OR. 2003. www.mentor.com.

"Designing USB Into Embedded Systems," SoftConnex Technologies, Irvine, CA. 2003. www.softconnex.com.

Enhanced Host Interface Controller Specification for Universal Serial Bus. Intel. www.intel.com/technology/usb/ehcispec.htm

Hyde, John, "USB Design by Example: A Practical Guide to Building I/ODevices," 2nd Edition, Intel Press, Santa Clara, CA.

OpenHCI — Open Host Controller Interface Specification for USB. Hewlett Packard. h18000.www1.hp.com/productinfo/development/openhci.html

Universal Host Controller Interface (UHCI) Design Guide. Intel. ftp://download.intel.com/design/usb/UHCI11D.PDF.

USB Implementers' Forum, www.usb.org. Lots of useful and background information.

Editor's Note: If you have added a USB port to a non-Windows product, I'd like to hear from you (in confidence) at jontitus@comcast.net.